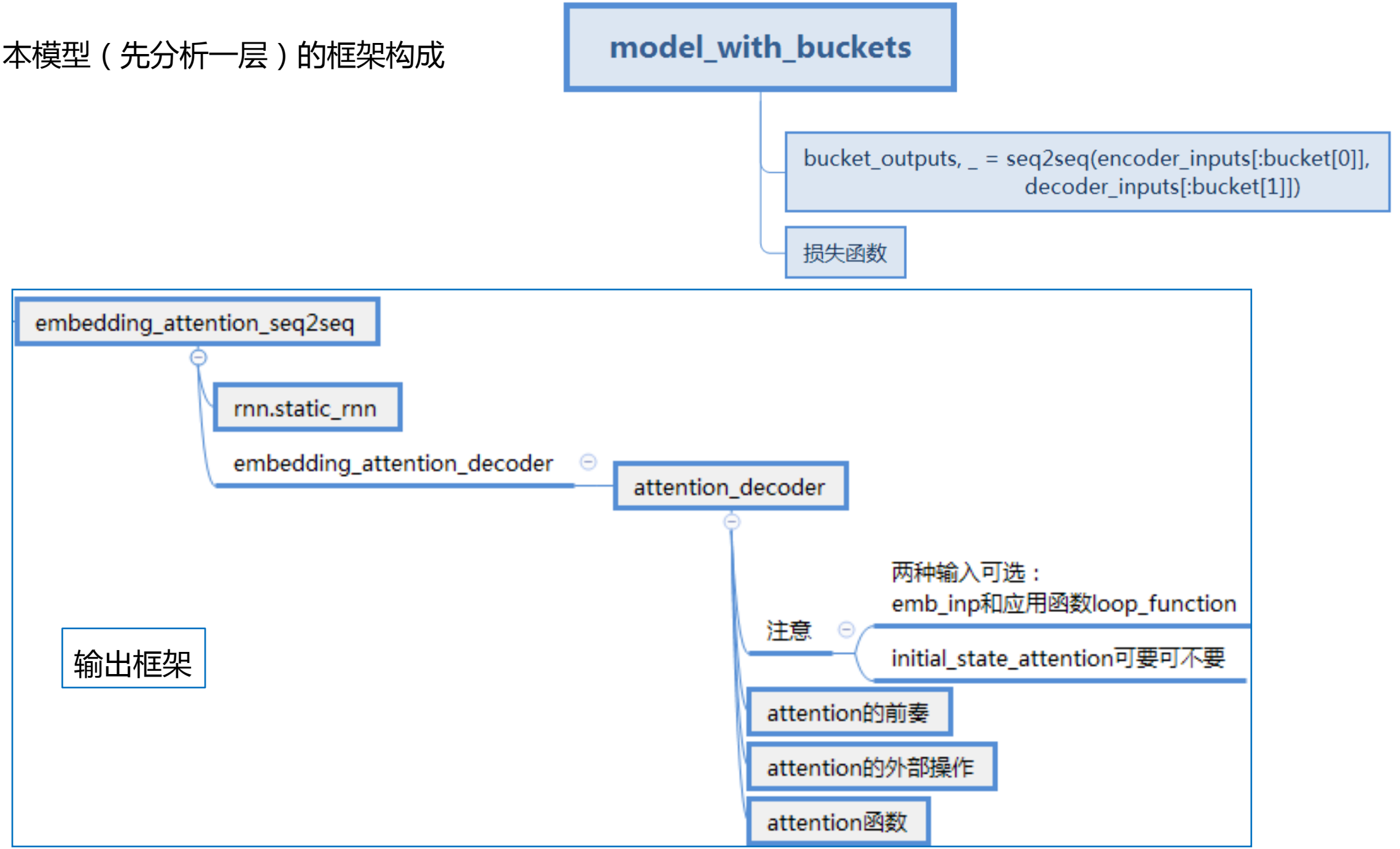
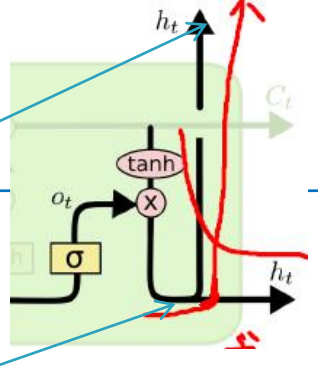
一、本模型（先分析一层）的框架构成



二、Encoder单元



attn_length：字数；

attn_size：字的维度

```
# Encoder.
encoder_cell = copy.deepcopy(cell)
encoder_cell = core_rnn_cell.EmbeddingWrapper(
    encoder_cell,
    embedding_classes=num_encoder_symbols,
    embedding_size=embedding_size)
encoder_outputs, encoder_state = rnn.static_rnn(
    encoder_cell, encoder_inputs, dtype=dtype)
# First calculate a concatenation of encoder outputs to put attention on.
top_states = [
    array_ops.reshape(e, [-1, 1, cell.output_size]) for e in encoder_outputs
]#增加1维，变为#【batchsize，字数，字维度】
attention_states = array_ops.concat(top_states, 1)#几个字合并成一个向量
```

副产品 V

```
# 为了计算W1 * h_t，我们用了1个 1-by-1的卷积核，该操作前需要对数据先进行reshape.
hidden = array_ops.reshape(attention_states,
                           [-1, attn_length, 1, attn_size])
hidden_features = []
v = []
attention_vec_size = attn_size  # Size of query vectors for attention.维度大小没变
for a in xrange(num_heads):
  k = variable_scope.get_variable("AttnW_%d" % a,
                                  [1, 1, attn_size, attention_vec_size])
  hidden_features.append(nn_ops.conv2d(hidden, k, [1, 1, 1, 1], "SAME"))
  v.append(
      variable_scope.get_variable("AttnV_%d" % a, [attention_vec_size]))
state = initial_state#（即前面传入的 encoder_state ）
```

三、Decoder单元

2.1、attention的前奏：输入

输入inp有两种，当条件具备时，优先选择prev：

```
for i, inp in enumerate(decoder_inputs):
  if i > 0:
    variable_scope.get_variable_scope().reuse_variables()
  # If loop_function is set, we use it instead of decoder_inputs.
  if loop_function is not None and prev is not None:
    with variable_scope.variable_scope("loop_function", reuse=True):
      inp = loop_function(prev, i)#inp被取代了
```

Attens的初始化有两种，当条件具备时，优先选择函数化initial_state:

```
attns = [
    array_ops.zeros(
        batch_attn_size, dtype=dtype) for _ in xrange(num_heads)
]
for a in attns:  # 确保the second shape of attention vectors is set.
  a.set_shape([None, attn_size])
if initial_state_attention:
  attns = attention(initial_state)
```

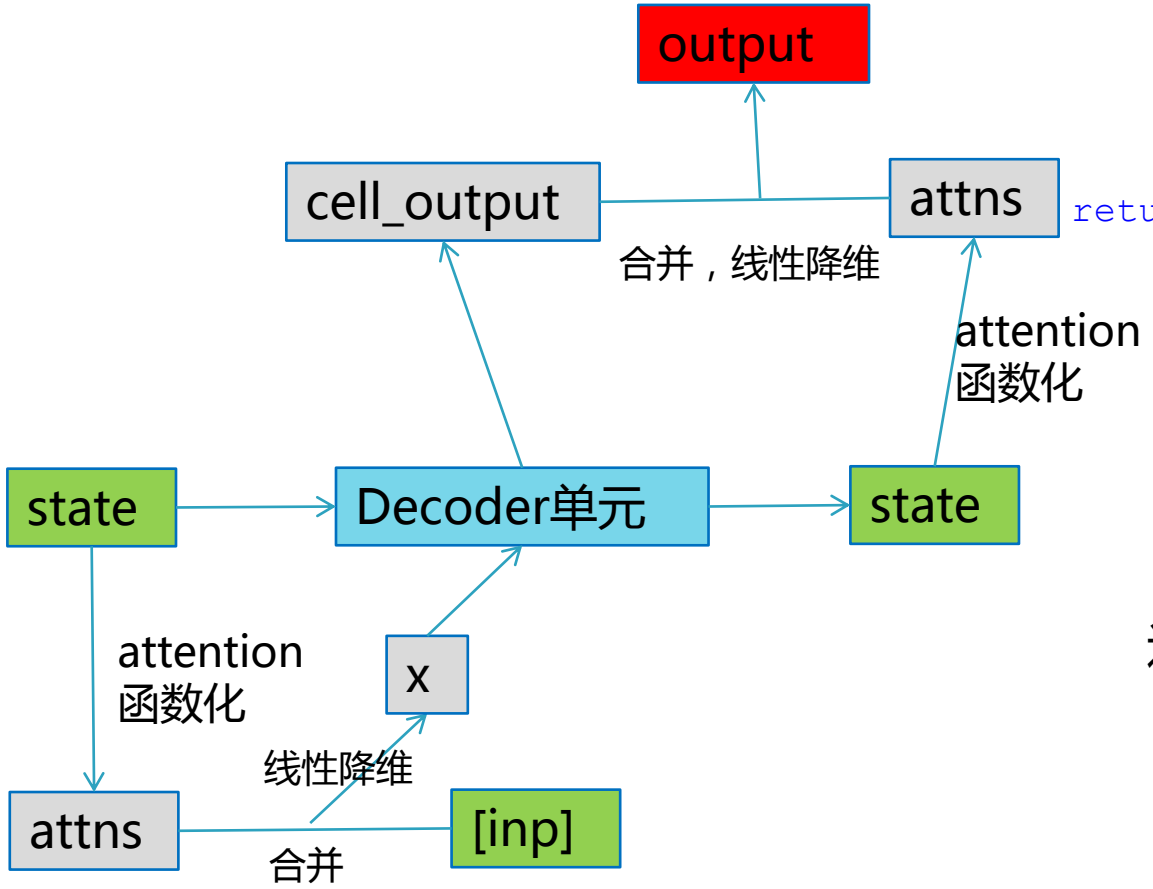
inputs = [inp] + attns

合并input and previous attentions 为 one vector of the right size.

x = Linear(inputs, input_size, True)(inputs)#先叠加，再线性降维，【1024,512】

三、Decoder单元

2.2、attention函数的外部操作：加性attention




```
# Run the RNN.
cell_output, state = cell(x, state)
# Run the attention mechanism.
if i == 0 and initial_state_attention:
  with variable_scope.variable_scope(
      variable_scope.get_variable_scope(), reuse=True):
    attns = attention(state)
else:
  attns = attention(state)
with variable_scope.variable_scope("AttnOutputProjection"):
  inputs = [cell_output] + attns
  output = Linear(inputs, output_size, True)(inputs)
if loop_function is not None:
  prev = output
outputs.append(output)
return outputs, state
```

（1）Attention 如果用一句话来描述，那就是：

encoder 层的输出经过**加权平均**后再输入到 decoder 层中。

这个加权可以用矩阵来表示，也叫 Attention 矩阵。

三、Decoder单元

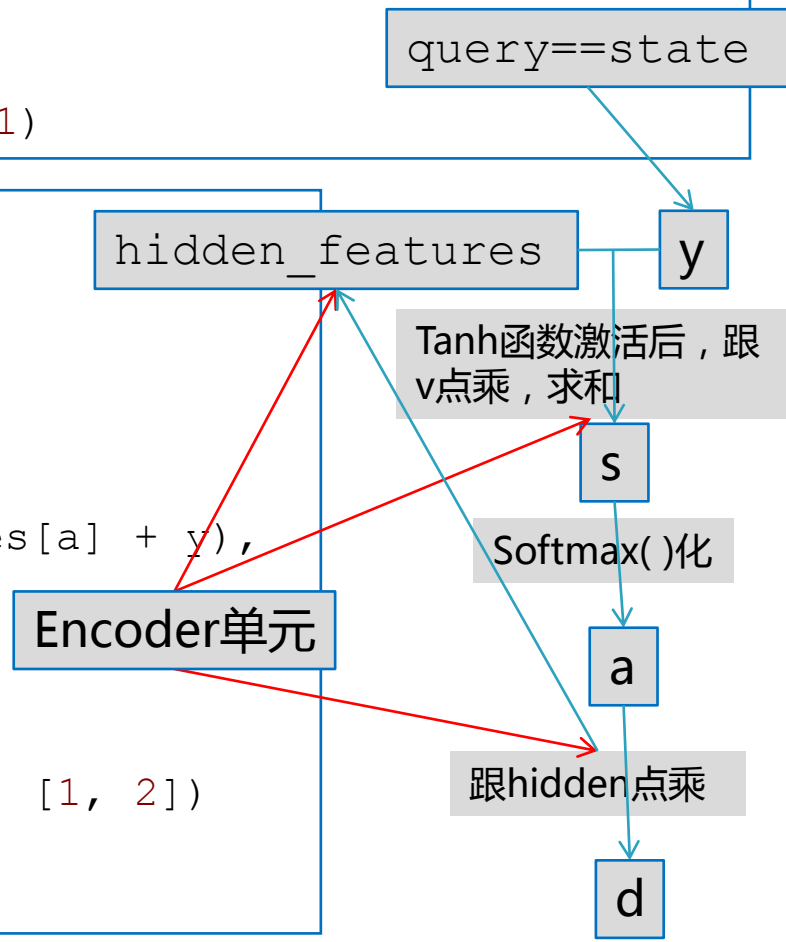
2.3、attention函数的内部机制

Attention矩阵：表示对于某个时刻的输出 y，它在输入 x 上各个部分的注意力。

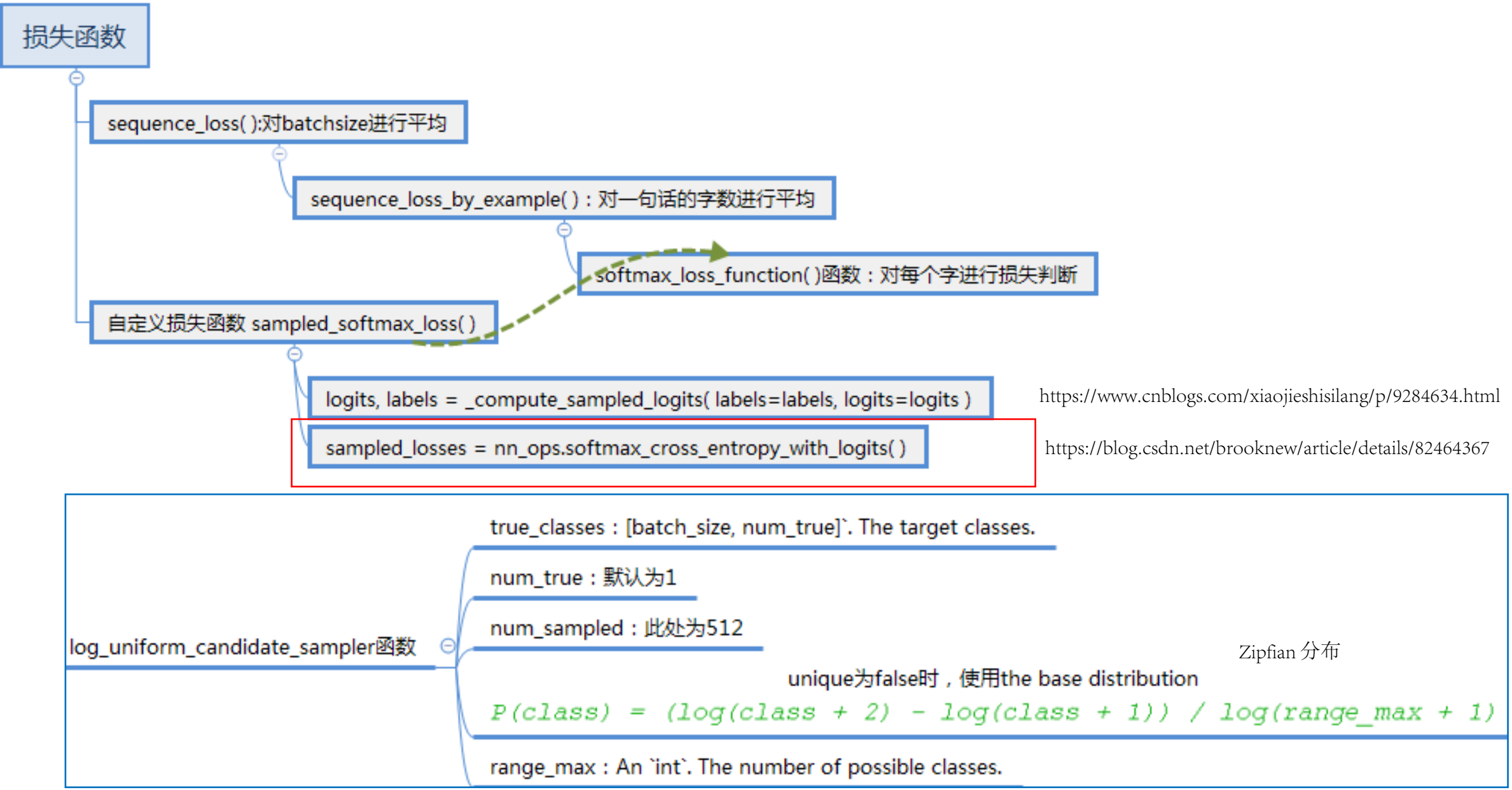
（1）预处理 query ：两层LSTM的state，flatten成一个列表；检验其维度个数后，合并

```
if nest.is_sequence(query):  # If the query is a tuple, flatten it.
  query_list = nest.flatten(query)
  for q in query_list:  # Check that ndims == 2 if specified.
    ndims = q.get_shape().ndims
    if ndims:
      assert ndims == 2
  query = array_ops.concat(query_list, 1)
```

```
ds = []
for a in xrange(num_heads):
  with variable_scope.variable_scope("Attention_%d" % a):
    y = Linear(query, attention_vec_size, True)(query)#线性降维
    y = array_ops.reshape(y, [-1, 1, 1, attention_vec_size])
    # Attention mask is a softmax of v^T * tanh(...).
    s = math_ops.reduce_sum(v[a] * math_ops.tanh(hidden_features[a] + y),
                            [2, 3])
    a = nn_ops.softmax(s)
    # Now calculate the attention-weighted vector d.
    d = math_ops.reduce_sum(
        array_ops.reshape(a, [-1, attn_length, 1, 1]) * hidden, [1, 2])
    ds.append(array_ops.reshape(d, [-1, attn_size]))
return ds
```



四、损失函数


损失函数
sequence_loss( ):对batchsize进行平均
sequence_loss_by_example( ) : 对一句话的字数进行平均
softmax_loss_function( )函数 : 对每个字进行损失判断
自定义损失函数 sampled_softmax_loss( )
logits, labels = _compute_sampled_logits(labels=labels, logits=logits)
sampled_losses = nn_ops.softmax_cross_entropy_with_logits( )
https://www.cnblogs.com/xiaojieshisilang/p/9284634.html
https://blog.csdn.net/brooknew/article/details/82464367
log_uniform_candidate_sampler函数
true_classes : [batch_size, num_true]`. The target classes.
num_true : 默认为1
num_sampled : 此处为512
unique为false时，使用the base distribution
P(class) = (log(class + 2) - log(class + 1)) / log(range_max + 1)
range_max : An `int`. The number of possible classes.
Zipfian 分布

五、补充

（1）loop_function(prev, i)函数

```
#Get a loop_function that extracts the previous symbol and embeds it.
def loop_function(prev, _):
    if output_projection is not None:
      prev = nn_ops.xw_plus_b(prev, output_projection[0], output_projection[1])
    prev_symbol = math_ops.argmax(prev, 1)
    # Note that gradients will not propagate through the second parameter of
    # embedding_lookup.
    emb_prev = embedding_ops.embedding_lookup(embedding, prev_symbol)
    if not update_embedding: #只有训练的时候才更新
      emb_prev = array_ops.stop_gradient(emb_prev)
    return emb_prev
```

W'过程：

（1）隐含层通过 N*V 维的权重矩阵 W' 连接到输出层。这样看来，W'其实只是一个升维的投影过程！

（2）将其结果作为softmax分类过程的输入来得到y值。在word2vec中我们采取argmax( )来得到每行（词）的ids，在词汇表中一找，就确定这个词了。